ARTIC WOLF: SAVING THE FAMILY

David Agudelo Ochoa

Erika Dayana León Quiroga

Universidad de Antioquia
Despartamento de Ingeniería Electrónica y Telecomunicaciones
Informática II
Medellín-Antioquia
Abril de 2021

Índice

1. Sección introductoria.

En este informe veremos el desarrollo del proyecto final de la materia Informática II, el cual será un juego llamado Gartic Wolf: Saving the family. Dentro de las secciones veremos listas de tareas hechas para organizar mejor el trabajo a realizar, actualizaciones y problemas del código del programa.

2. Lista de tareas.

- Jueves 7 de Abril:
  - Crear proyecto Qt
  - Crear la clase ObjetoAnimado y las que heredarán de ella.
  - Objeto animado: Atributos y métodos.
  - Método StringPath.
  - Probar método animar.
  - Instalar librería QMediaPlayer (para sonidos del juego.)
- Viernes 8 de Abril:
  - Agregar atributos y constructor de las clases que heredan de Objeto animado, como héroe, enemigo, proyectil, reloj.
  - Creación y animación del escenario y el héroe.
  - Pruebas de la animación.
- Sábado 9 de Abril:
  - Agregar métodos de las clases que heredan de Objeto animado, como héroe, enemigo, proyectil, reloj.
  - Método de proyectil que mueve y verifica la colisión con otros objetos del mismo.
  - Agregar el método que creará el nivel uno.
- Domingo 10 de Abril:
  - Buscar ecuaciones de péndulo.
  - Conectar las vidas, y el tiempo con la finalización del nivel.
  - Agregar el menú.
  - Personalizar el menú.
- Lunes 11 de Abril:
  - Modelar el comportamiento del péndulo.
  - Terminar nivel uno.
  - Tareas faltantes:
    - Asignar botones de comandos.
    - Diseñar la imagen del menú con los comandos previamente establecidos.
    - Diseñar pantallas de información, como las intrucciones, la historia. Y pantallas de carga como el cambio de nivel.
    - Implementar el poder del héroe.
- Martes 12 de Abril:
  - Empezar a trabajar en el guardado del juego.

- Ecuaciones para el resorte y el movimiento parabólico.
- Modelar proyectil de nivel dos.

- Miércoles 13 de Abril:
  - Agregar movimiento del jugador en el nivel dos.
  - Prueba de la jugabilidad del nivel dos.
- Jueves 14 de Abril:

3. Desarrollo del proyecto.

Para el desarrollo del proyecto Artic Wolf: Saving the Family, se tuvo en cuenta el informe de Clases que se realizó previamente con la intención de tener un plan de organización de las clases que serían necesarias para el funcionamiento del juego que se tiene planeado. En este informe de clases encontramos la descripción nivel por nivel del juego y la descripción clase por clase del mismo. A continuación veremos los cambios que se le realizaron a las clases que se tenían pensadas a la hora de materializar estas ideas en código.

3.1. Jueves 7 y Viernes 8 de Abril.

Se comenzó por crear el proyecto con la clase padre Objeto animado con sus respectivos atributos y métodos, y sus clases hijas Heroe, Enemigo, Proyectil y Reloj. En la clase objeto animado se realizaron varios cambios con respecto a lo planeado en el informe de clases. Esto porque a medida que se va escribiendo el código se encuentran formas más efectivas de realizar ciertas tareas que las que ya se tenían planeadas. Sin embargo, el uso del informe de Clases es fundamental para la asignación de tareas y la realización del avance del proyecto.

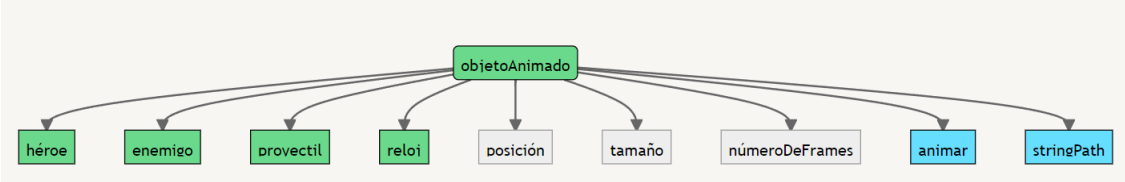

Figura 1: Clase ObjetoAnimado que se tenía pensada construir.

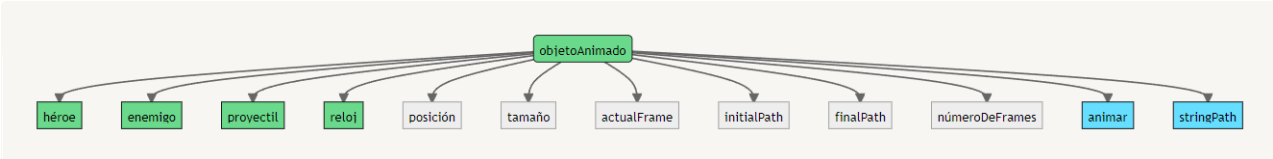

Figura 2: Clase ObjetoAnimado construida.

Con los cambios realizados en la clase objeto animado se logró obtener el movimiento esperado en los objetos que se muestran en la escena, esto con ayuda del método animar y stringPath los cuales siguen cumpliendo las mismas funciones que se les asignaron en la parte de planeación del proyecto. El método stringPath recibe el número de la imagen de la cuál se requiere su path y luego lo retorna. La función animar utiliza el método anterior para conocer el camino a la imagen y haciendo uso de actualFrame como parámetro del método stringPath logra ir cambiando el frame de tal forma que cuando sea llamada, estos varíen y den la impresión de movimiento.

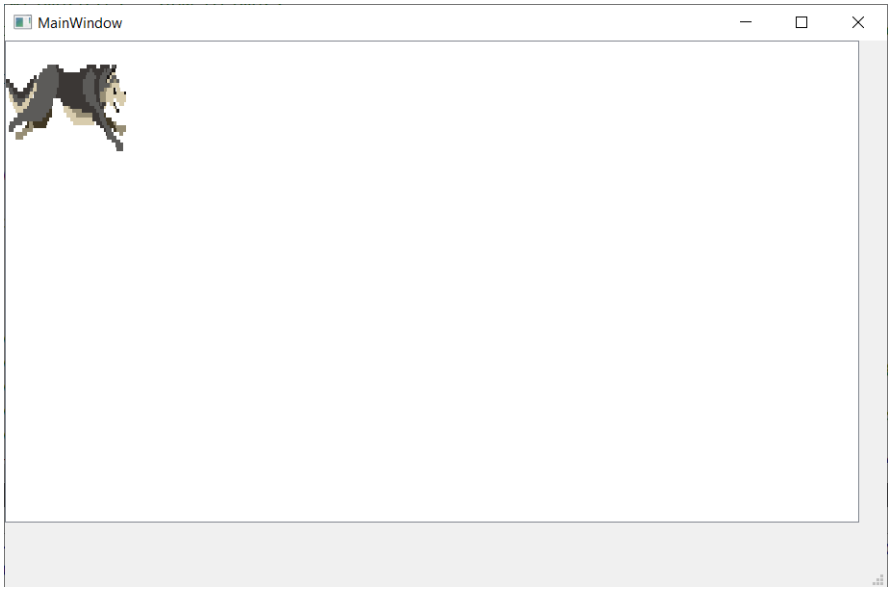

Figura 3: Pruebas del funcionamiento de la clase objeto animado.

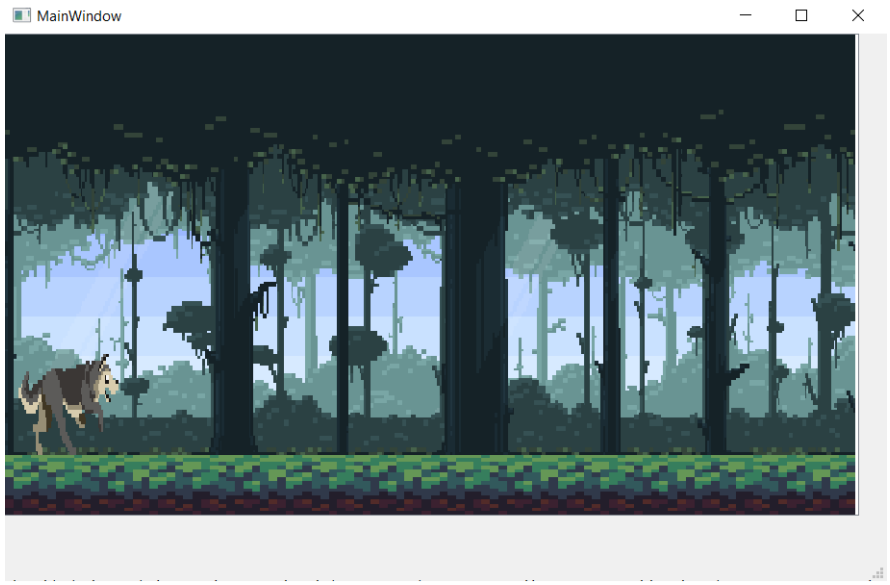

Figura 4: Pantalla de prueba con el héroe y el fondo animados.

3.2. Sábado 9 de Abril.

Se empezó con la creación de la clase padre Nivel y sus clases hijas nivel1 y nivel2 como se planeó en el informe de Clases. Estas clases se utilizarían para armar el nivel correspondiente en cada clase.

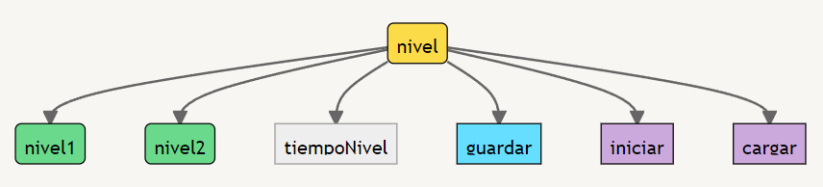

Figura 5: Clase nivel planeada en el informe de clases.

Sin embargo, nos dimos cuenta que la función que le otorgamos a estas clases en la planeación podría ser reemplazada por un **método en el MainWindow** que realice la misma función, en este método se agregarían todos los objetos necesarios para crear el nivel correspondiente. Por lo tanto la clase padre nivel y sus hijas son descartadas.

Este mismo día se comienza con el modelamiento del proyectil del nivel 1, para esto utilizamos el planteamiento de la clase proyectil que se realizó en el informe de clases.

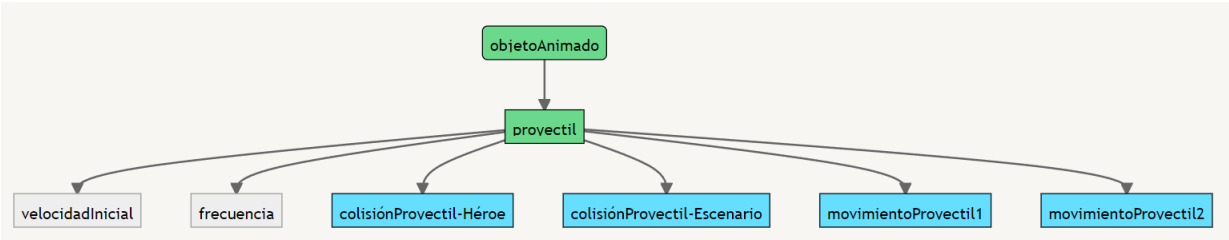

Figura 6: Clase nivel planeada en el informe de clases.

Se trabaja en la colisión entre el proyectil y el escenario, tratando de que cada que el proyectil salga de este se borre de la escena y se cree un nuevo proyectil, pero el borrado de este objeto provocaba conflictos en el proyecto por lo que se decidió que no se borraría el objeto sino que se le haría una especie de Reset a su posición, de esta forma no se estarían creando y borrando nuevos objetos sino que siempre es el mismo, y cuando sale de la escena o se choca con el héroe vuelve a su posición inicial. El movimiento automático del proyectil se logra haciendo uso de un Timer y conectándolo con las ecuaciones de movimiento del proyectil del nivel 1.

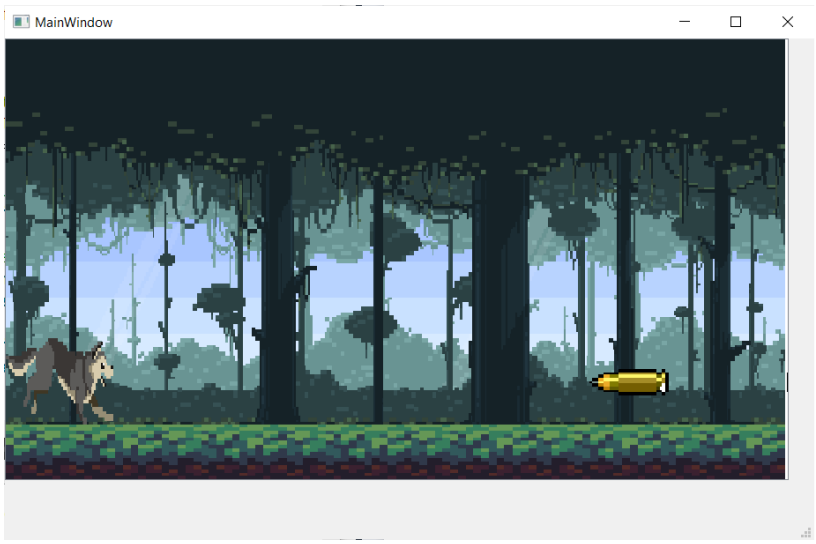

Figura 7: Pruebas de la animación y la colisión entre el proyectil y la pared.

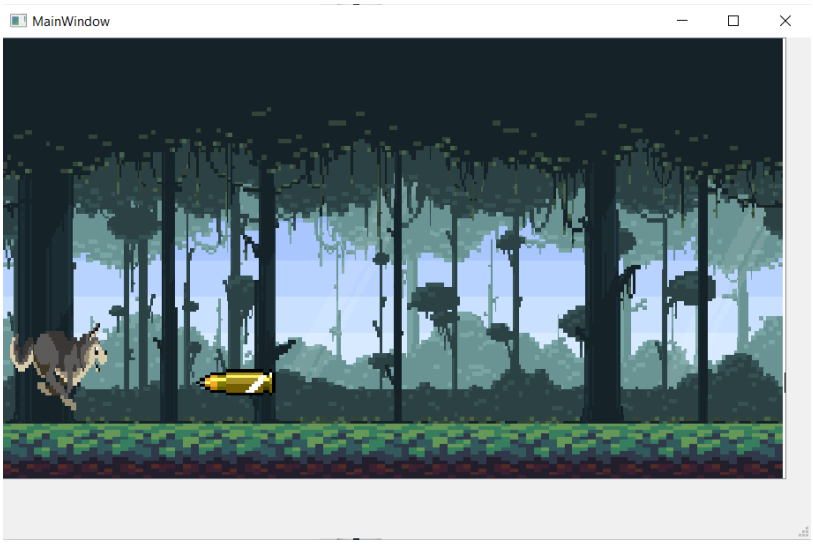

Figura 8: Pruebas de la animación y la colisión entre el proyectil y la pared.

3.3. Domingo 10 de Abril.

Se agregó la animación del enemigo, se ajustó la ventana de GraphicsView y el tamaño de todos los objetos en la escena.

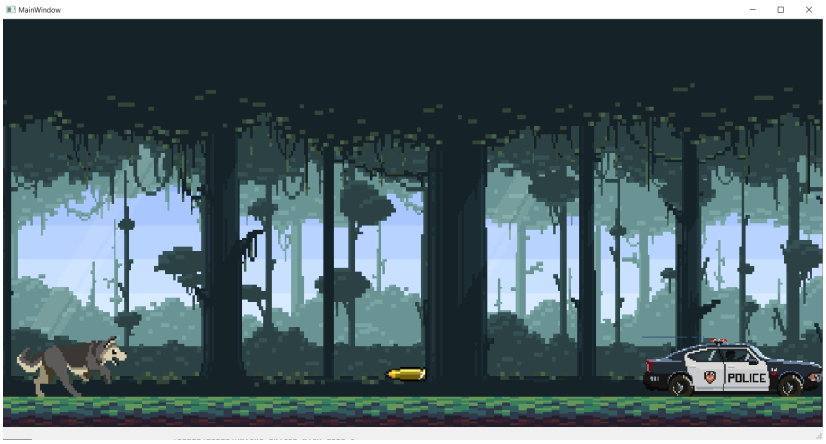

Figura 9: Ajuste del tamaño de los objetos de la escena.

También se agregó el contador de las vidas y del tiempo de la partida como podremos ver en la siguiente imagen (10), estos contadores se ubicarán en la parte superior izquierda de la escena, en la imagen de demostración están encerrados en un recuadro rojo para ubicarlos de la manera más sencilla. El tiempo irá disminuyendo conforme transcurre la partida con ayuda de un Timer. Las vidas disminuirán cada que un proyectil choque con el héroe y esto será codificado en el método de la clase proyectil en donde se verifica la colisión entre el héroe y el proyectil.

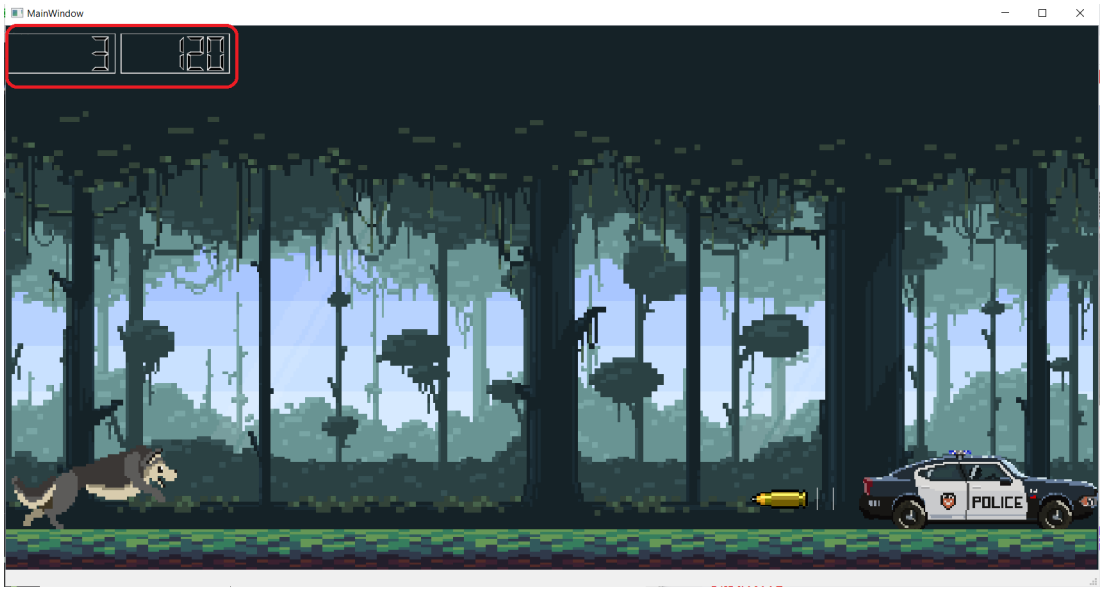

Figura 10: Reloj y vidas del héroe puestas en pantalla.

Se hicieron pruebas para agregar la pantalla del menú, la cuál aparecerá cuando se corra el juego y una vez se presione la tecla I del teclado iniciará la partida con tres vidas y 120 segundos en el contador del tiempo de la partida. Esto se hará con un KeyPressEvent y un if que al presionar la tecla determinada iniciará el método del MainWindow encargado de crear los objetos del nivel 1.

Figura 11: Pantalla que mostrará el menú.

```
if(i->key() == Qt::Key_I){
    cargarNivel1();
}
```

Figura 12: Condicional encargado de iniciar el nivel 1 cuando se presione la tecla I.

PROBLEMAS: La animación del héroe tiene problemas en el momento en el que el héroe se encuentra realizando un salto. Esto se debe a que la animación del enemigo se encuentra dentro del SLOT de animación del héroe para ahorrar líneas de código.
POSIBLES SOLUCIONES: Una de las posibles soluciones es meter la animación del enemigo dentro de otro SLOT de animación que ya exista, como el del fondo o el proyectil para de esta forma seguir ahorrando líneas de código. La otra opción es simplemente crear un nuevo SLOT que se encargue de la animación del enemigo.

3.4. Lunes 11 de Abril.

Se comenzó solucionando el problema que se venía presentando con la animación del enemigo, se probaron las dos posibles soluciones planteadas y se tomó la decisión de crear un nuevo SLOT para la animación del enemigo así esto conllevara líneas extras de código. Luego de que el problema anterior fue solucionado se procedió a realizar el modelamiento del péndulo y la descripción de este movimiento con ayuda de un diagrama de cuerpo libre y las ecuaciones.

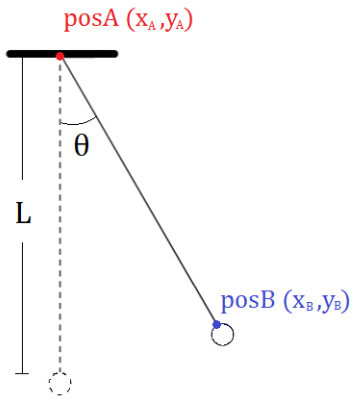

Figura 13: Diagrama de cuerpo libre del péndulo.

Para esto se agregaron nuevos atributos a la clase del reloj, como la longitud del péndulo, sus coordenadas en los puntos A y B y el ángulo inicial del mismo.

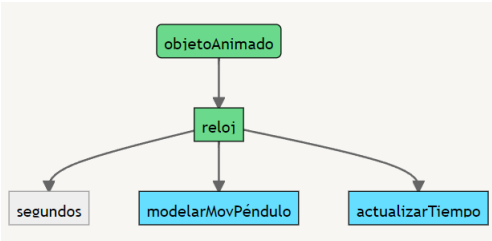

Figura 14: Clase reloj pensada en el informe de Clases.

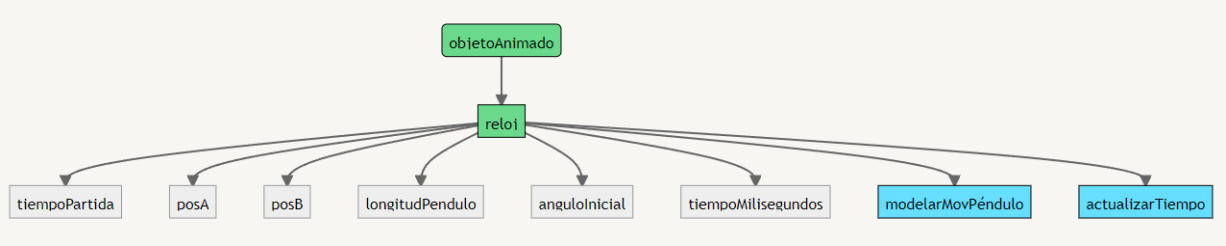

Figura 15: Clase reloj necesaria para la creación del péndulo.

También se creó un constructor diferente para el reloj, en el que se incluyen los valores iniciales necesarios para la construcción del péndulo, estos valores serán las coordenadas del punto A, la longitud del péndulo y el ángulo inicial y con los valores anteriormente mencionados se obtendrán las coordenadas iniciales del punto B. Esto, como ya se mencionó, serán los valores de las coordenadas iniciales del péndulo.

$$x_B = x_A + Lsin(\theta_0)$$
$$y_B = y_A + Lcos(\theta_0)$$

Figura 16: Ecuaciones para determinar las coordenadas iniciales del punto B.

Luego, para realizar el movimiento pendular se necesitará ir cambiando las coordenadas en B con respecto avance el tiempo, para esto se necesitará el ángulo, que también irá cambiando con respecto al tiempo, un omega que está relacionado con la longitud del péndulo, la longitud del péndulo y las coordenadas en A.

$$\omega = \sqrt{\frac{g}{L}}$$
$$\theta = \theta_0 \cos(\omega t)$$
$$x_B = x_A + Lsin(\theta)$$
$$y_B = y_A + Lcos(\theta)$$

Figura 17: Ecuaciones que determinan el movimiento del péndulo. Siendo g la gravedad, L la longitud del péndulo, y theta sub cero el ángulo inicial.

Después de realizar la implementación de las ecuaciones del péndulo y verificar su funcionamiento, se agregó la imagen del reloj y del péndulo para la ambientación del juego.

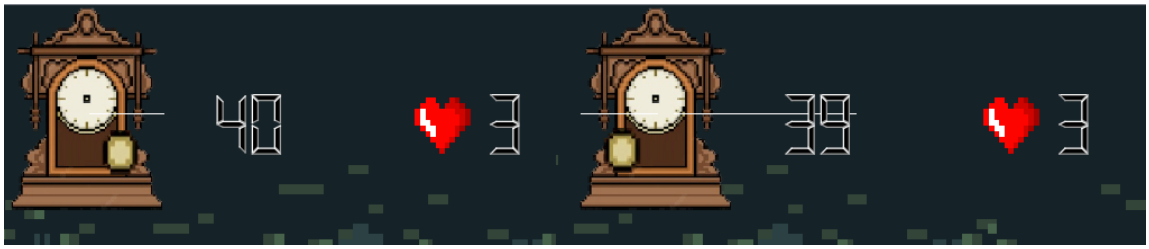

Figura 18: Imagen del reloj y péndulo funcionando.

Este mismo día se comenzó con la prueba de las ecuaciones de movimiento parabólico que serán utilizadas en el nivel 2 para el lanzamiento del proyectil.

3.5. Martes 13 de Abril.

Como se tenía planeado se comenzó a abordar el problema del guardado y la carga del juego, para comenzar se tendrán en cuenta los aspectos más importantes del juego, como lo son el número de vidas del héroe, el tiempo que lleva la partida y el nivel en el que se encuentra a la hora de guardar. Para esto se creó una nueva clase que no estaba contemplada en la planeación de las clases, pues en esta planeación se tenía pensado incluir el guardado y cargado en la clase de niveles, clase que finalmente no fue implementada en el código, por esto se creó una clase que se encarga de guardar y cargar las partidas cuando el usuario así lo requiere. Para implementar la clase se utilizó la práctica tres como guía del manejo de archivos, pues es utilizando estos que podremos guardar y cargar la partida.

Para guardar la partida creamos un método que tendrá como parámetros su nombre, y punteros a objetos de tipo reloj y de tipo héroe, lo anterior para poder utilizar los métodos Get del tiempo de partida, las vidas del héroe y el nivel en el que se encuentra, que son los datos relevantes del juego. Los datos anteriores se escribirán en el archivo haciendo un salto de línea en cada uno para su fácil lectura cuando se desee cargar el juego. Para cargar la partida se codificará otro método con los mismos parámetros del método de guardado, con la diferencia que el archivo de este método será de lectura, para tomar los datos guardados en el archivo de escritura y cargar el nivel con estos datos.

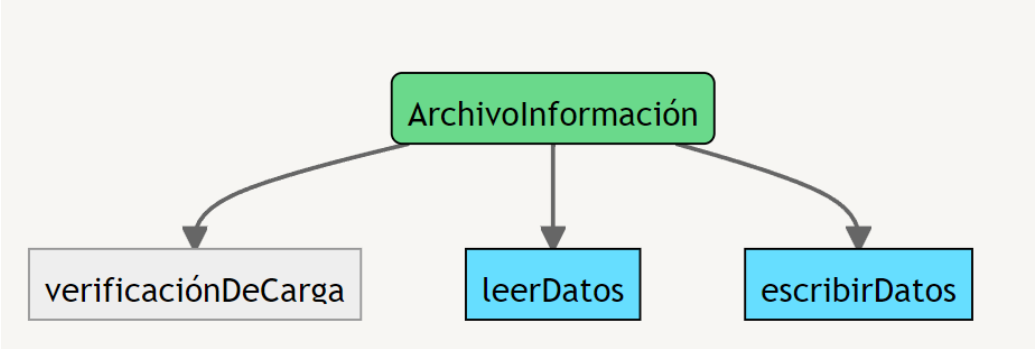

Figura 19: Mapa mental de la clase que manejará el guardado y cargado del juego.

En este día también se implementó el Movimiento Parabólico haciendo uso de sus ecuaciones. Estas ecuaciones se utilizarán para el movimiento del proyectil del nivel dos.

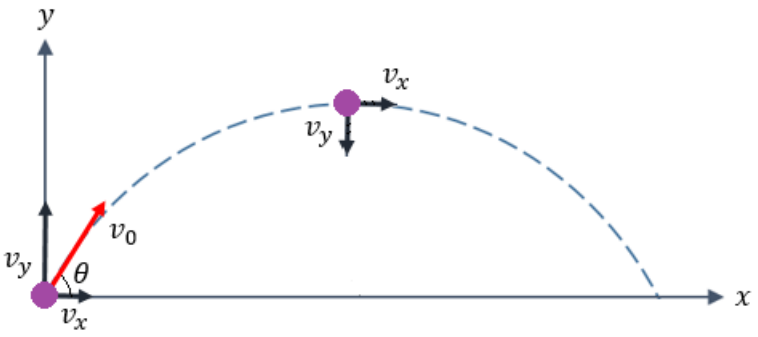

Figura 20: Diagrama de cuerpo libre del movimiento parabólico.

$$V_x = V_0 \cos(\theta) \qquad x = x_0 + V_x t$$

$$V_y = V_0 \sin(\theta) - gt \qquad y = y_0 + V_y t + \frac{1}{2} g t^2$$

$$V = \sqrt{V_x + V_y}$$

$$\theta = \tan^{-1} \frac{V_y}{V_x}$$

Figura 21: Ecuaciones del movimiento parabólico.

También implementamos las ecuaciones de resortes, las cuales nos darán la velocidad inicial del proyectil que variará dependiendo de la compresión que se le realice al resorte, esta compresión será aleatoria. Para todas las ecuaciones usadas en este proyecto se tuvo que tener en cuenta el sistema de referencia de la escena en donde se agregan todos los objetos necesarios para el funcionamiento del juego.

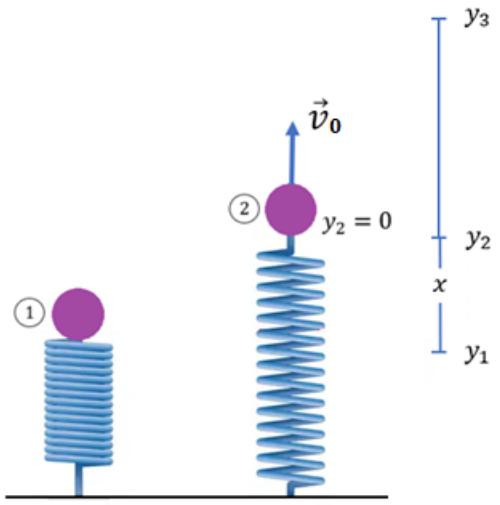

Figura 22: Diagrama de cuerpo libre del resorte.

$$y_1 = y_2 - \mathrm{x}$$

$$v = \sqrt{2gy_1 + \frac{kx^2}{m}}$$

Figura 23: Ecuaciones para la velocidad inicial dada por un resorte.